OPEN SOURCE ENGINEERING

open eering

A SCILAB PROFESSIONAL PARTNER

powered by ENGIN SOFT

PLOTTING IN SCILAB

In this Scilab tutorial we make a collection of the most important plots arising in scientific and engineering applications and we show how straightforward it is to create smart charts with Scilab.

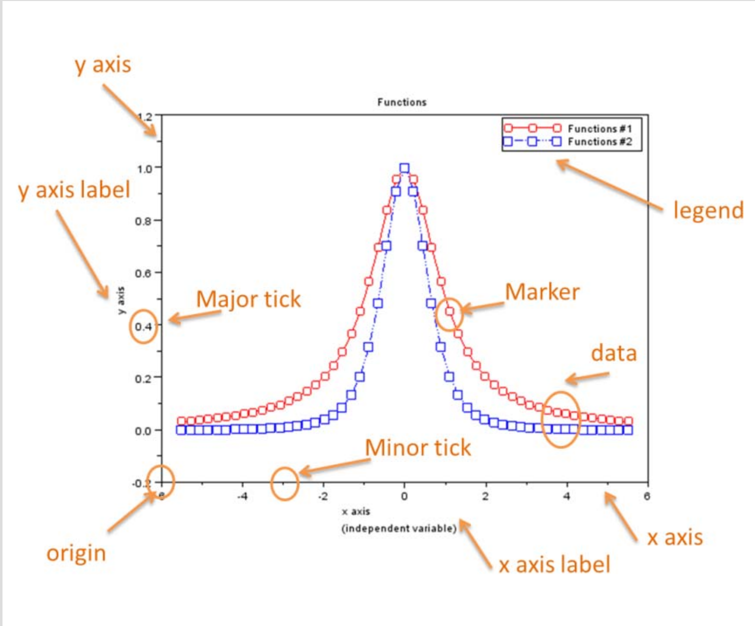
Level

www.openeering.com

Step 1: The purpose of this tutorial

The purpose of this Scilab tutorial is to provide a collection of plotting examples that can be used in Scilab to show data.

Here, on the right, we report some definitions used when plotting data on figures.



Step 2: Roadmap

Examples refer to 1D, 2D, vector fields and 3D problems. In this Scilab tutorial, the reader will discover some basics commands on how to add annotations in LaTex, manage axis, change plotting properties such as colors, grids, marker size, font size, and so on.

This tutorial can be considered as a ***quick kick-start guide*** for engineers and scientists for ***data visualization in Scilab***.

Descriptions	Steps
One dimensional plot	1-7
Bi-dimensional plot	8-12
Tri-dimensional plot	13-14
Animated plot	15

Step 1: Basic plot with LaTex annotations

Here, we plot the function:

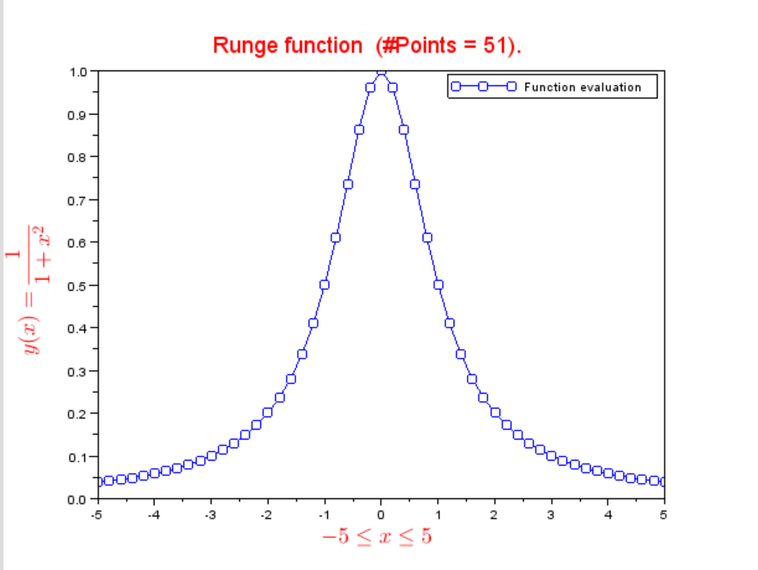
$$y = \frac{1}{1+x^2}$$

on the interval $[-5,5]$.

```
// Close all opened figures and clear workspace
xdel(winsid());
clear;
clc;

// Figure #1: Basic plot with LaTex annotations
// ----------
// Data
x = linspace(-5,5,51);
y = 1 ./(1+x.^2);

// Plot
scf(1);
clf(1);
plot(x,y,'o-b');
xlabel("$-5\le x\le 5$","fontsize",4,"color","red");
ylabel("$y(x)=\frac{1}{1+x^2}$","fontsize",4,"color","red");
title("Runge function  (#Points = 
"+string(length(x))+").","color","red","fontsize",4);
legend("Function evaluation");
```



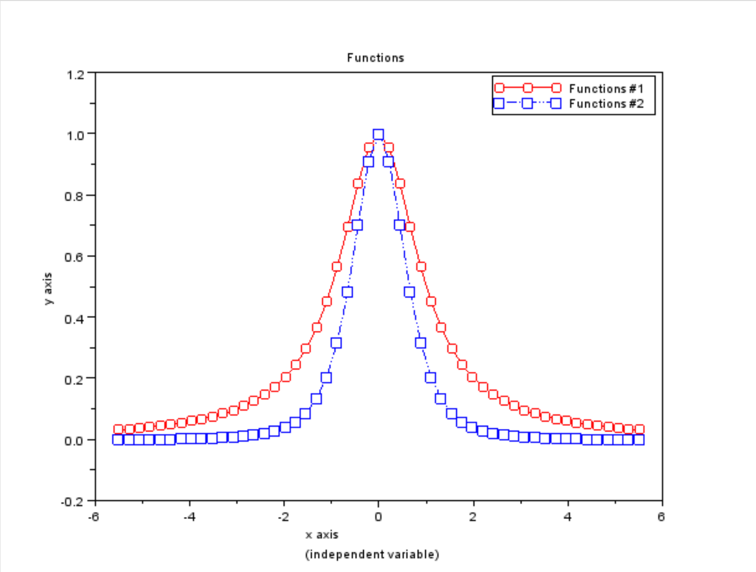
Step 2: Multiple plot and axis setting

In this example we plot two functions on the same figure using the command ***plot*** twice.

Then, we use the command ***legend*** to add an annotation to the figure.

With the command ***gca*** we get the handle to the current axes with which it is possible to set axis bounds.

```
// Figure #2: Multiple plot and axis setting
// ----------
// Data
x = linspace(-5.5,5.5,51);
y = 1 ./(1+x.^2);
// Plot
scf(2);
clf(2);
plot(x,y,'ro-');
plot(x,y.^2,'bs:');
xlabel(["x axis";"(independent variable)"]);
ylabel("y axis");
title("Functions");
legend(["Functions #1";"Functions #2"]);
set(gca(),"data_bounds",matrix([-6,6,-0.1,1.1],2,-1));
```

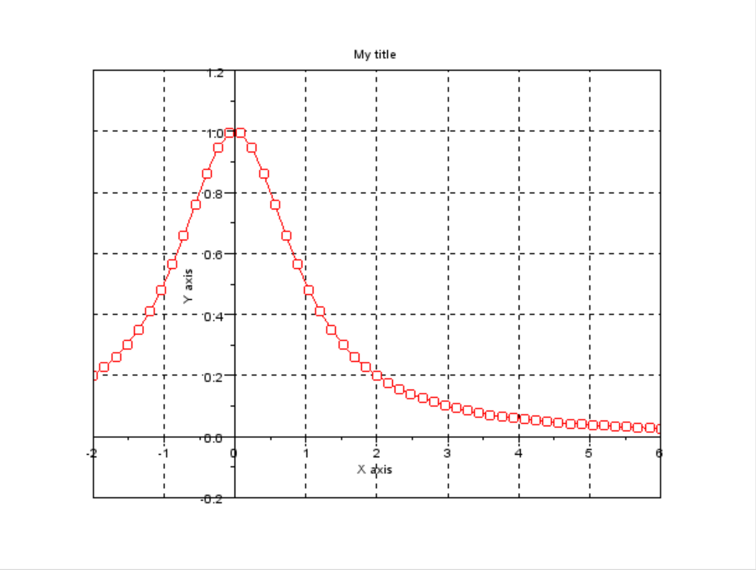


Step 3: Change axis origin and add grid

In this example the function is plotted over a grid with axis are located inside the figure.

The command ***gca*** is used to get an handle to the figure axis and, hence, to access the axis fields.

```
// Figure #3 : Change axis origin and add grid
// -----------
// Data
x = linspace(-2,6,51);
y = 1 ./(1+x.^2);
// Plot
scf(3);
clf(3);
plot(x,y,'ro-');
set(gca(),"grid",[1 1]);
a = gca(); // get the current axes
a.x_location = "origin";
a.y_location = "origin";
set(gca(),"data_bounds",matrix([-2,6,-0.2,1.2],2,-1));
xtitle("My title", "X axis", "Y axis");
```

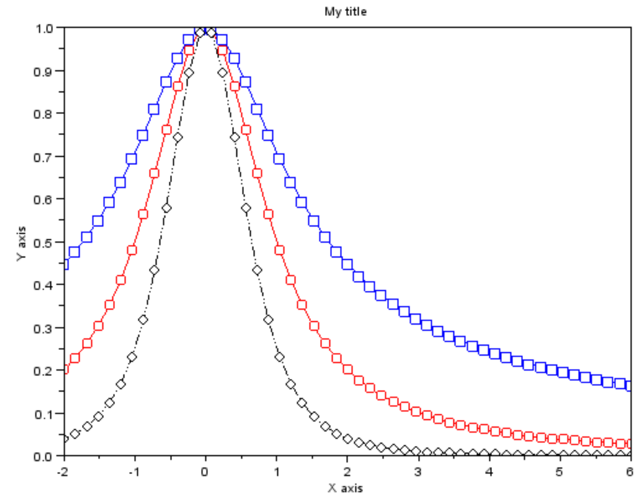


Step 4: Another example of multiple plot

This is another way to manage multiple plot. Please notice that the color black is denoted by ***'k'*** since the letter ***'b'*** is for the color blue.

To MATLAB® users this command may recall ***"hold on"*** and ***"hold off",*** just be careful that the concept of "on" and "off" are here reversed.

```
// Figure #4 : Another example of multiple plot
// -----------
// Data
x = linspace(-2,6,51);
y = 1 ./(1+x.^2);
// Plot
scf(4);
clf(4);
set(gca(),"auto_clear","off")
plot(x,y,'ro-');
plot(x,sqrt(y),'bs-');
plot(x,y.^2,'k:d');
set(gca(),"auto_clear","on")
xtitle("My title", "X axis", "Y axis");
```



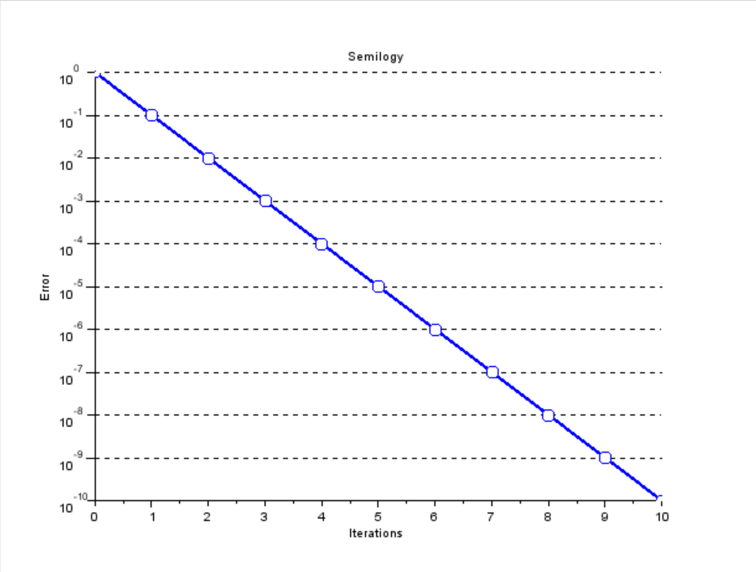
Step 5: Semilogy plot

When you have small values to show (e.g. errors, convergence data) a semilogy plot is mandatory.

The log axis is assigned with the command `plot2d("nl",…)`. The string "`nl`" indicates that the first axis is "***normal***" and the second axis is "***logarithmic***".

If the string is reversed ("`ln`") we have a plot with a logarithmic scale in the x and a normal scale in the y.

```
// Figure #5 : Semilogy plot
// -----------
// Data
iter = linspace(0,10,11);
err = 10.^(-iter);
// Plot
scf(5);
clf(5);
plot2d("nl", iter, err, style=2);
p = get("hdl");
p.children.mark_mode = "on";
p.children.mark_style = 9;
p.children.thickness = 3;
p.children.mark_foreground = 2;
xtitle("Semilogy", "Iterations", "Error");
set(gca(),"grid",[-1 1]);
```



Step 6: Loglog plot

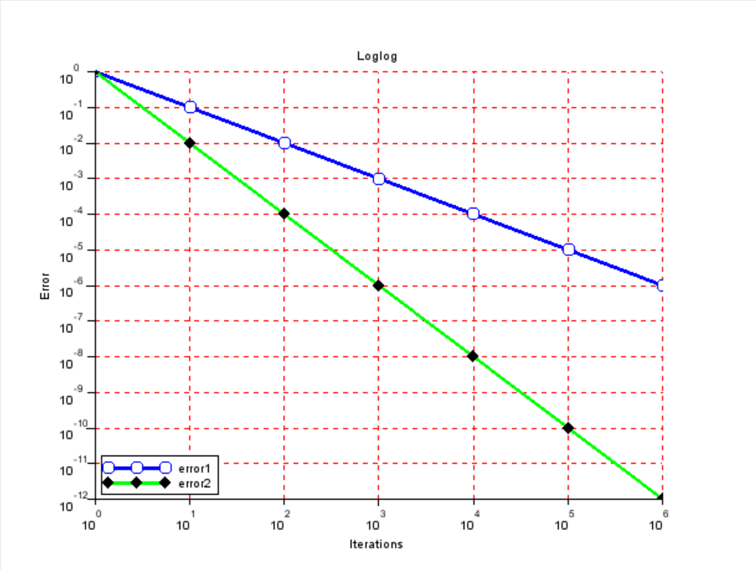
This plot is logarithmic on both axis as the reader can recognize from the figure on the right (see labels and ticks) and from the command `plot2d("ll",…)`.

This kind of charts is widely used in electrical engineering (e.g. for radio frequency) when plotting gain and filter losses and whenever decibels (dB) versus frequencies should be plotted. A famous log-log plot is the Bode diagram.

```
// Figure #6 : Loglog plot
// -----------
// Data
ind = linspace(0,6,7);
iter = 10.^ind;
err1 = 10.^(-ind);
err2 = (10.^(-ind)).^2;
// Plot
scf(6);
clf(6);
plot2d("ll", iter, err1, style=2);
p = get("hdl");
p.children.mark_mode = "on";
p.children.mark_style = 9;
p.children.thickness = 3;
p.children.mark_foreground = 2;

plot2d("ll", iter, err2, style=3);
p = get("hdl");
p.children.mark_mode = "on";
p.children.mark_style = 4;
p.children.thickness = 3;
p.children.mark_foreground = 1;

xtitle("Loglog", "Iterations", "Error");
set(gca(),"grid",[5 5]);
// legend(['error1';'error2'],"in_upper_left");
legend(['error1';'error2'],"in_lower_left");
```

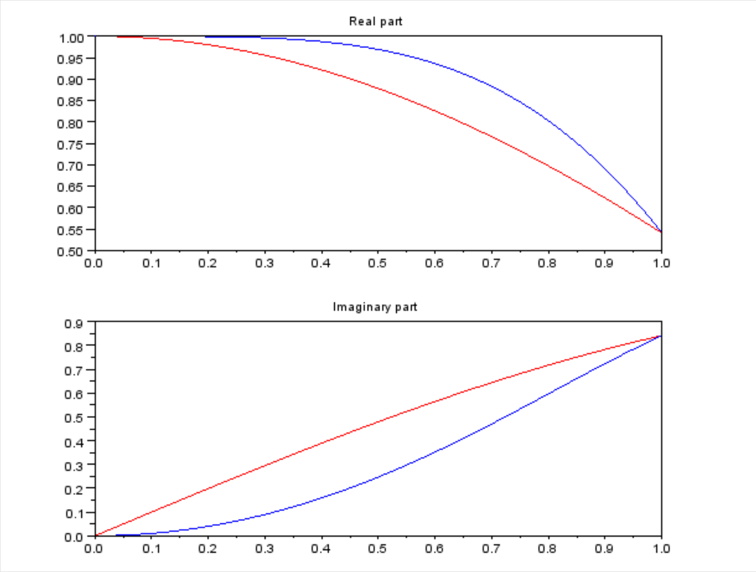


Step 7: Subplot with real and imaginary part

In this figure we have two plots in the same chart: ***real*** and ***complex***.

This kind of plotting is particularly useful in signal processing, control theory and many other fields. For example, with this chart we can plot magnitude and phase of a Fast Fourier Transform (FFT) analysis.

```
// Figure #7 : Subplot with real and imaginary part
// -----------
// Data
t = linspace(0,1,101);
y1 = exp(%i*t);
y2 = exp(%i*t.^2);
// Plot
scf(7);
clf(7);
subplot(2,1,1);
plot(t,real(y1),'r');
plot(t,real(y2),'b');
xtitle("Real part");
subplot(2,1,2);
plot(t,imag(y1),'r');
plot(t,imag(y2),'b');
xtitle("Image part");
```



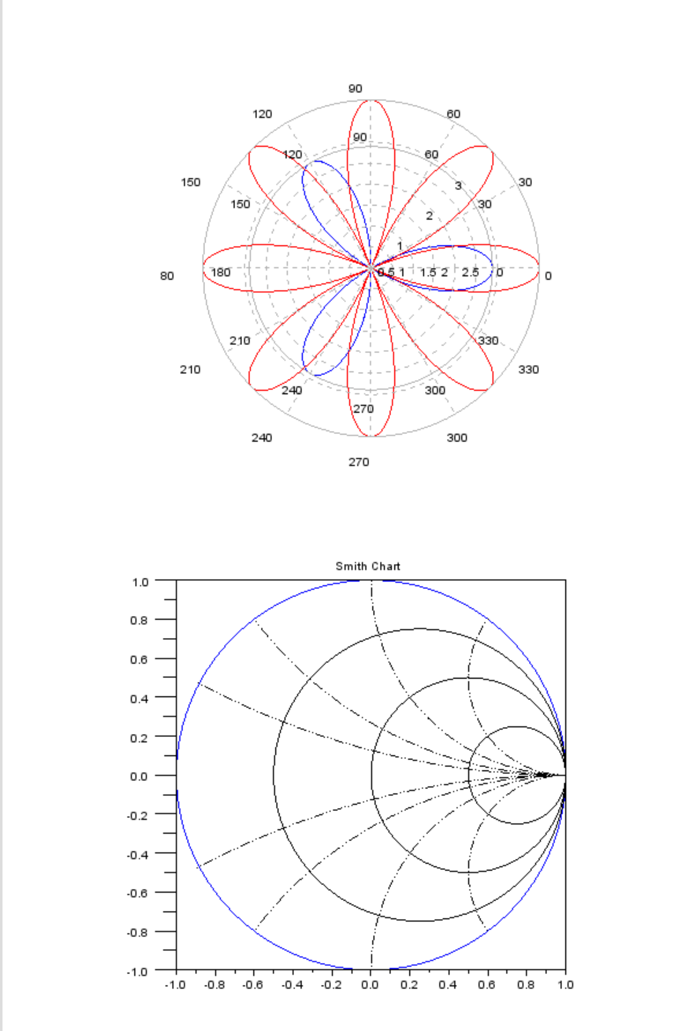
Step 8: Radial plot

Radial charts are important for displaying phenomena characterized by direction and distance from a fixed point, such as temperature distribution in the Earth. These phenomena have a ***cyclic structure*** in some directions.

A typical application occurs in Radio Frequency (RF) engineering where the Smith chart for scattering parameters, and for impedance transmission coefficients is widely used.

```
// Figure #8 : Radial plot
// -----------
// Data
theta = 0:.01:2*%pi;
a = 1.7; r1 = a^2*cos(3*theta);
a = 2; r2 = a^2*cos(4*theta);
// Plot
scf(8); clf(8);
subplot(2,1,1);
polarplot(theta,r1,[2,2]);
polarplot(theta,r2,[5,2]);

// Smith Chart
subplot(2,1,2);
t = linspace(0, 2*%pi, 101);
x = cos(t); y = sin(t);
plot(x, y);
k = [.25 .5 .75]';
x1 = k*ones(t) + (1 - k)*cos(t);
y1 = (1 - k)*sin(t);
plot(x1', y1', 'k');
kt = [2.5 %pi 3.79 4.22]; k = [.5 1 2 4];
for i = 1:length(kt)
    t = linspace(kt(i), 1.5*%pi, 51);
    a = 1 + k(i) * cos(t);
    b = k(i) + k(i) * sin(t);
    plot(a, b,'k:', a, -b,'k:');
end
a = gca(); a.isoview = 'on';
xtitle("Smith Chart");
set(gca(),"data_bounds",matrix([-1,1,-1,1],2,-1));
```

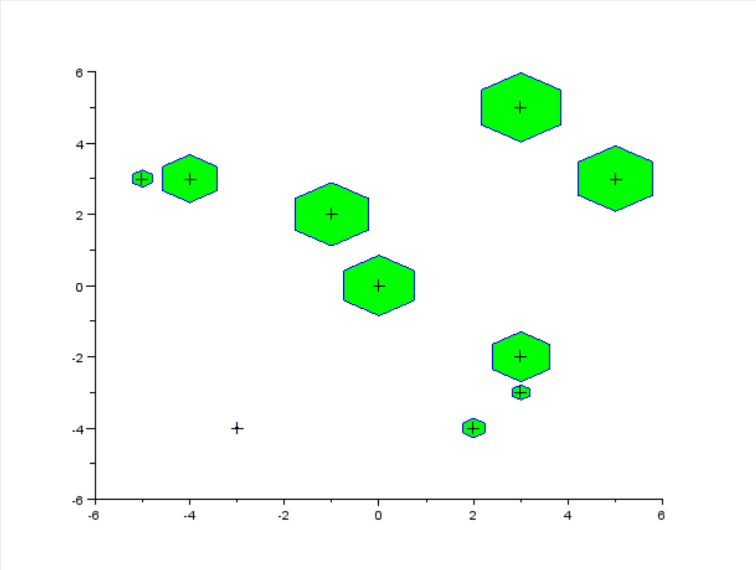


Step 9: Polygon plot

This chart represents a customization of the famous ***bubble chart*** in which bubbles are substituted by a ***polygon*** (in this case an hexagon). The function that describes the polygon may be re-written to form any kind of parametric shape.

As for bubble charts, these polygon plots are particularly useful to represent three distinct parameters on a two-dimensional chart.

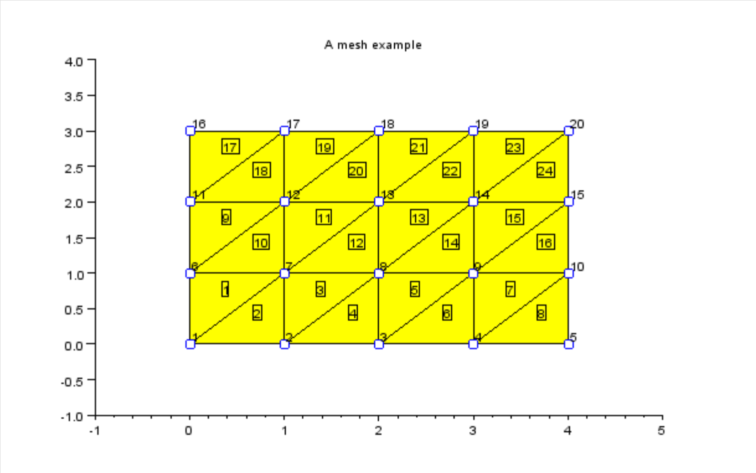
```
// Figure #8 : Polygon plot
// -----------
// Data
deff('[x,y]=hexagon(c,r)',['x = c(1) + r*sin(2*%pi*(0:5)/6);','y
= c(2) + r*cos(2*%pi*(0:5)/6);']);
n = 10;
xy = round((rand(n,2)-0.5)*10);
rr  = round(rand(n,1)*100)/100;
// Plot
scf(9);
clf(9);
for i=1:length(rr)
    c = xy(i,:);
    r = rr(i);
    [x,y]=hexagon(c,r);
    xpoly(x,y,"lines",1);
    e=gce(); // get the current entity
    e.fill_mode ="on"
    e.foreground = 2;
    e.background = 3;
end
plot2d(xy(:,1),xy(:,2),-1);
set(gca(),"data_bounds",matrix([-6,6,-6,6],2,-1));
```



Step 10: Mesh example

In finite elements method (FEM) plotting a mesh is essential. This code show a simple regular mesh with its node and triangular enumerations.

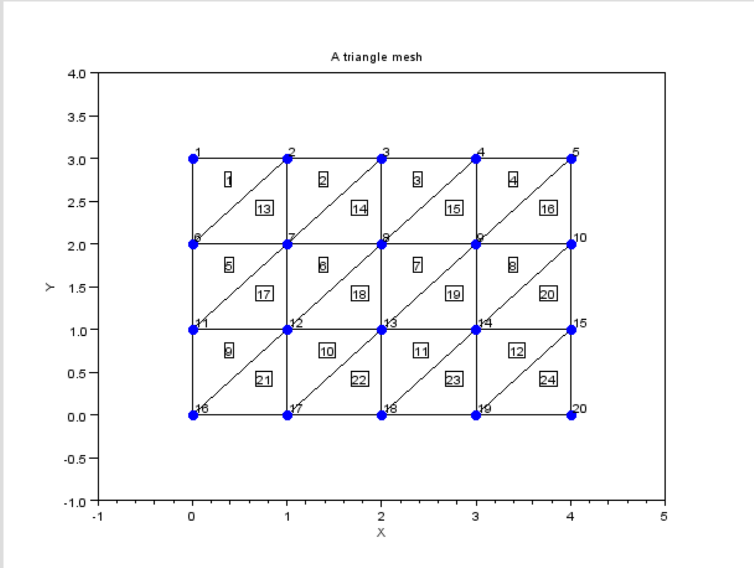
```
// Figure #9 : Mesh example
// -----------
// Data: Node coordinate matrix
coord_lx = [0 1 2 3 4]; coord_x = repmat(coord_lx,1,4);
coord_ly = [0 0 0 0 0]; coord_y = [];
for i=1:4, coord_y = [coord_y, coord_ly+i-1]; end
coord = [coord_x;coord_y]';
// Data: Connectivity matrix
inpoel = zeros(4*3*2,3); index = 1;
for j=1:3,
    for i=1:4,
        ind1 = i+(j-1)*5; ind2 = (i+1)+(j-1)*5;
        ind3 = (i+1)+j*5; ind4 = i+j*5;
        inpoel(index,:) = [ind1,ind3,ind4];
        inpoel(index+1,:) = [ind1,ind2,ind3];
        index = index + 2;
    end
end
// Data: some derived data
np = size(coord,1); nt = size(inpoel,1);
xtn = coord(inpoel,1); ytn = coord(inpoel,2);
xn = coord(:,1); yn = coord(:,2);
xtn = matrix(xtn, nt, length(xtn)/nt)';
xtrimesh = [xtn; xtn($,:)];
ytn = matrix(ytn, nt, length(ytn)/nt)';
ytrimesh = [ytn; ytn($,:)];
xbar = mean(xtn,'r'); ybar = mean(ytn,'r');
// Plot
scf(10); clf(10);
xfpolys(xtrimesh,ytrimesh,repmat(7,1,nt));
plot(xn,yn,'bo');
xstring(xn,yn,string(1:np));
xstring(xbar,ybar,string(1:nt),0,1);
set(gca(),"data_bounds",matrix([-1,5,-1,4],2,-1));
xtitle("A mesh example");
```



Step 11: Another mesh example

```
// Figure #10 : Another mesh example
// ------------
xp = [0 1 2 3 4 0 1 2 3 4 0 1 2 3 4 0 1 2 3 4];
yp = [3 3 3 3 3 2 2 2 2 2 1 1 1 1 1 0 0 0 0 0];

coord = [xp',yp'];
intma = [1 6 2; 2 7 3; 3 8 4; 4 9 5;
         6 11 7; 7 12 8; 8 13 9; 9 14 10;
        11 16 12; 12 17 13; 13 18 14; 14 19 15;
         6 7 2; 7 8 3; 8 9 4; 9 10 5;
        11 12 7; 12 13 8; 13 14 9; 14 15 10;
        16 17 12; 17 18 13; 18 19 14; 19 20 15;];
// baricentric coordinates
it1 = intma(:,1); it2 = intma(:,2); it3 = intma(:,3);
xbar = (xp(it1)+xp(it2)+xp(it3))/3;
ybar = (yp(it1)+yp(it2)+yp(it3))/3;
np = length(xp);
nt = size(intma,1);
// plot mesh
vertex = coord;
face = intma;
xvf =
matrix(vertex(face,1),size(face,1),length(vertex(face,1))/size(fa
ce,1))';
yvf =
matrix(vertex(face,2),size(face,1),length(vertex(face,1))/size(fa
ce,1))';
zvf =
matrix(zeros(vertex(face,2)),size(face,1),length(vertex(face,1))/
size(face,1))';
// Plotting
tcolor = repmat([0.5 0.5 0.5],nt,1);
scf(11); clf(11);
plot3d(xvf,yvf,list(zvf,tcolor));
xtitle("A triangle mesh");
a = gca();
a.view = "2d";
a.data_bounds=[min(xp)-1,min(yp)-1;max(xp)+1,max(yp)+1];
// plot node
plot(xp,yp,'.')
// xstring(xp,yp,string((1:np)'));
xnumb(xp,yp,1:np); xnumb(xbar,ybar,1:nt,[1]);
```



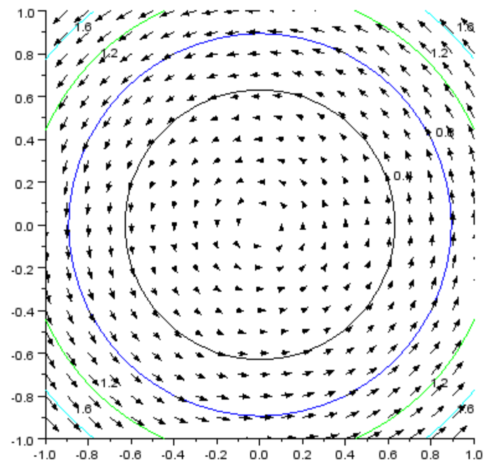
Step 12: Vector field and contour plot

A vector field displays velocity vectors as arrows with components (fx,fy) at the points (x,y).

Vector field charts are typically used in fluid-dynamics where it is necessary to visualized the velocity field.

In fluid-dynamics we encounter ***vector field*** combined with a ***contour*** plot for showing the velocity field and the pressure derived from solving Navier-Stokes equations.

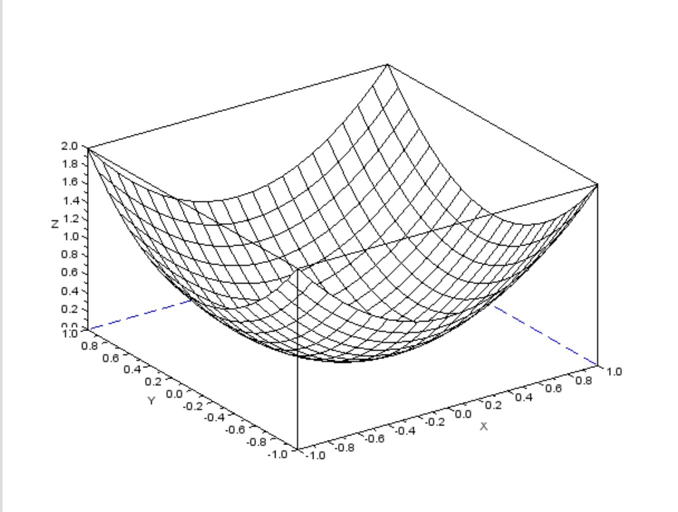
```
// Figure #14: Vector field and contour plot
// Data
x = -1:0.1:1; y = -1:0.1:1;
[X,Y] = meshgrid(x,y);
Z = X.^2 + Y.^2;
// Plot
scf(12);
clf(12);
contour(x,y,Z,4);
champ(x,y,-X,Y,rect=[-1,-1,1,1]);
a = gca();
a.isoview = 'on';
```



Step 13: Scilab mesh function

This chart is an example on how to plot a surface on a 3D chart. The command ***meshgrid*** is used to create a bi-dimensional grid. The function is then computed in that grid and finally plot with the command mesh.

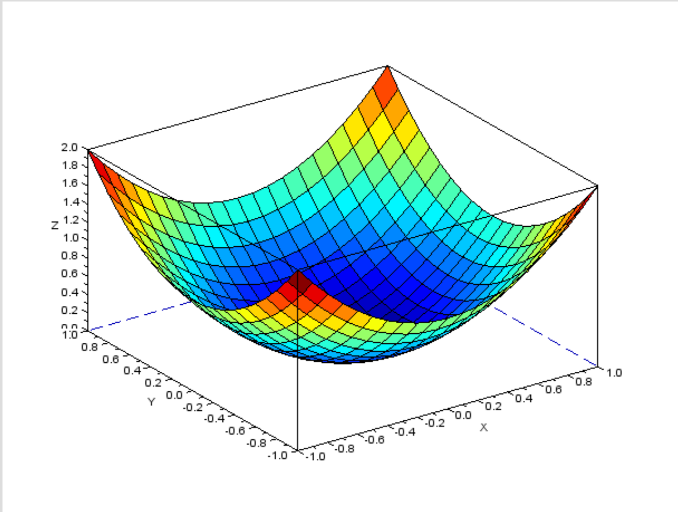
```
// Figure #12 : Mesh
// ------------
// Data
x = -1:0.1:1;
y = -1:0.1:1;
[X,Y] = meshgrid(x,y);
Z = X.^2 + Y.^2;
// Plot
scf(13);
clf(13);
mesh(X,Y,Z);
xlabel('X');ylabel('Y');zlabel('Z');
```



Step 14: Surface with a colormap

In this example we use the command ***surf*** with a color map.

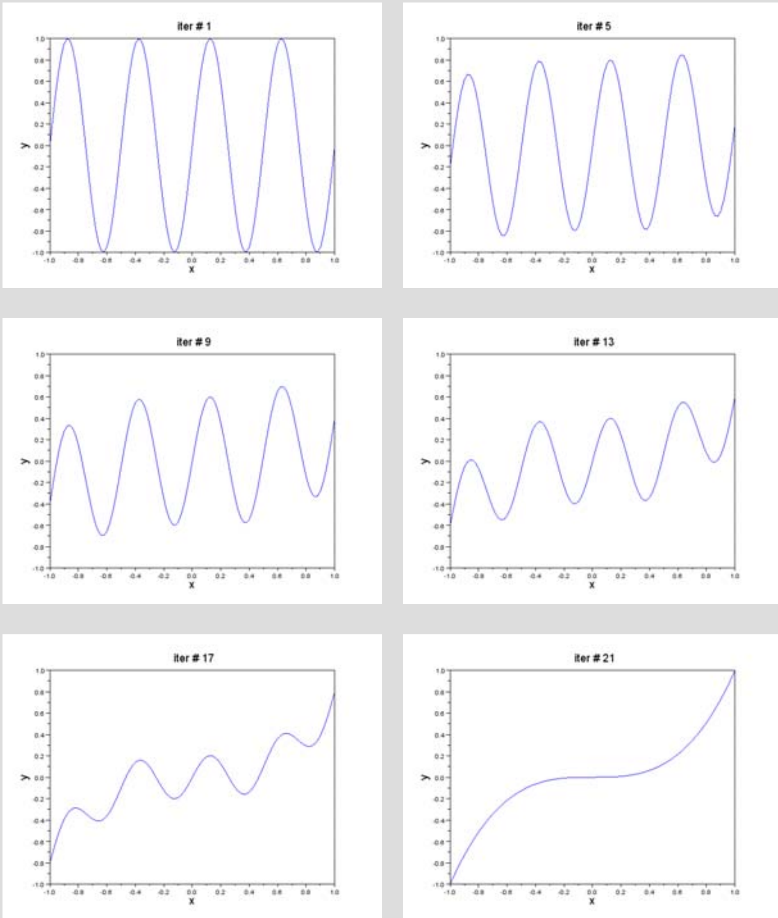
```
// Figure #11 : Surface with a colormap
// ------------
// Data
x = -1:0.1:1;
y = -1:0.1:1;
[X,Y] = meshgrid(x,y);
Z = X.^2 + Y.^2;
// Plot
scf(14);
clf(14);
xset("colormap",jetcolormap(64));
surf(X,Y,Z);
xlabel('X');ylabel('Y');zlabel('Z');
```



Step 15: Creating a video in Scilab

With the following command we produce a sequence of “png” images with can then be combined to generate a video or an animated png image.

```
// Data
x = linspace(-1,1,101);
kval = linspace(0,1,21);
deff('[y]=mymorph(x,k)','y=(1-k)*sin(2*%pi*2*x) + k*x.^3');
// Iterative plot
for i=1:length(kval)
    k = kval(i)
    y = mymorph(x,k);
    scf(15);
    clf(15);
    drawlater();
    plot(x,y,'b-');
    title(" iter # "+string(i),"fontsize",4);
    xlabel("x","fontsize",4);
    ylabel("y","fontsize",4);
    set(gca(),"data_bounds",matrix([-1,1,-1,1],2,-1));
    drawnow();
    // Save figure in png format
    xs2png(gcf(),sprintf("example_%03d.png",i));
    sleep(100);
end
```



The animated png movie can be done outside of this Scilab script starting from all saved images. To produce the movie, the reader can use his/her preferred available tool.

To create our animated png image we used the program ***JapngEditor***, which is free of charge and available for download (see reference).

Step 16: Concluding remarks and References

In this tutorial we have collected and explained a series of figures that are available in Scilab for plotting any charts that scientists and engineers may need in their daily work.

1. Scilab Web Page: Available: www.scilab.org.
2. Openeering: www.openeering.com.
3. JapngEditor : https://www.reto-hoehener.ch/japng/.

Step 17: Software content

To report a bug or suggest some improvement please contact Openeering team at the web site www.openeering.com.

Thank you for your attention,

Manolo Venturin

```
--------------
Main directory
--------------
plotting.sce                 : A collection of plot.
ex1_animated.png             : Example of animated png image
license.txt                  : The license file
```